EVIDEN

Identity and Access Management

DirX Directory

Containerization

Version 9.1, Edition June 2025

All product names quoted are trademarks or registered trademarks of the manufacturers concerned.

© 2025 Eviden

All Rights Reserved

Distribution and reproduction not permitted without the consent of Eviden.

Table of Contents

Preface

This document provides information about the packages available for creating containerized deployments of DirX Directory. It consists of the following chapters:

- Chapter 1 briefly describes each package and how it is intended to be used
- Chapter 2 describes the DirX Directory container image package
- Chapter 3 describes the example DirX Directory Kubernetes project package, including its structure and example workflows

DirX Directory Documentation Set

DirX Directory provides a powerful set of documentation that helps you configure your directory server and its applications.

The DirX Directory document set consists of the following manuals:

- *DirX Directory Introduction*. Use this book to obtain a description of the concepts of DirX Directory.
- *DirX Directory Administration Guide*. Use this book to understand the basic DirX Directory administration tasks and how to perform them with the DirX Directory administration tools.
- *DirX Directory Administration Reference*. Use this book to obtain reference information about DirX Directory administration tools and their command syntax, configuration files, environment variables and file locations of the DirX Directory installation.
- *DirX Directory Syntaxes and Attributes*. Use this book to obtain reference information about DirX Directory syntaxes and attributes.
- *DirX Directory LDAP Extended Operations*. Use this book to obtain reference information about DirX Directory LDAP Extended Operations.
- *DirX Directory External Authentication*. Use this book to obtain reference information about external authentication.
- *DirX Directory Supervisor*. Use this book to obtain reference information about the DirX Directory supervisor.
- *DirX Directory Plugins for Nagios*. Use this book to obtain reference information about DirX Directory plugins for Nagios.
- *DirX Directory Disc Dimensioning Guide*. Use this book to understand how to calculate and organize necessary disc space for initial database configuration and enhancing existing configurations.
- *DirX Directory Guide for CSP Administrators*. Use this book to obtain information about installing, configuring and managing DirX Directory in the context of a Certificate Provisioning Service operating in accordance with regulations like the German “Signaturgesetz”.
- *DirX Directory Release Notes*. Use this book to install DirX Directory and to understand the features and limitations of the current release.

Notation Conventions

Boldface type
In command syntax, bold words and characters represent commands or keywords that must be entered exactly as shown.

In examples, bold words and characters represent user input.

Italic type
In command syntax, italic words and characters represent placeholders for information that you must supply.

[ ]
In command syntax, square braces enclose optional items.

{ }
In command syntax, braces enclose a list from which you must choose one item.

In Tcl syntax, you must actually type in the braces, which will appear in boldface type.

|
In command syntax, the vertical bar separates items in a list of choices.

...
In command syntax, ellipses indicate that the previous item can be repeated.

install_path
The exact name of the root of the directory where DirX Identity programs and files are installed. The default installation directory is *userID_home_directory**/DirX Identity* on UNIX systems and **C:\Program Files\DirX\Identity** on Windows systems. During installation the installation directory can be specified. In this manual, the installation-specific portion of pathnames is represented by the notation *install_path*.

1. Overview

DirX provides two packages for DirX Directory containerization:

- An example container image of a full, standalone DirX Directory installation. This package can be deployed “as is” or used as a building block for creating shadowing and other configurations in a containerized environment.
- An example Kubernetes project that demonstrates the use and capabilities of the DirX Directory container image. This package can be used as a reference or development base when containerizing a custom DirX Directory installation.

This document assumes familiarity with DirX Directory, Kubernetes, and Docker terms, concepts, and tools and provides only the specifics for using the packages with each product. Using the packages described here requires at least a basic understanding of these products.

The documents listed below provide additional details about the concepts and procedures referenced in this document. We recommend that you become familiar with the information in these documents before proceeding with the tasks described in this document:

For DirX Directory:

DirX Directory Administration Reference:

- Environment Variables
- IP Port Numbers
- **dbaminit** command
- **dirxconfig** command

DirX Directory Administration Guide

- Setting up the DSA

For Kubernetes:

See the documentation available at https://kubernetes.io/docs/home

For Docker:

See the documentation available at https://docs.docker.com

2. Container Image

The DirX Directory container image is downloadable from the DirX support portal. This chapter describes the specifics of the DirX Directory container image.

2.1. Base Image

The DirX Directory container image is based on the latest opensuse/leap image hosted on **hub.docker.com**. As OpenSUSE is not a supported Linux platform, the delivered image is only intended to be used as an example in a test environment. For productive use, you must build your own docker image from one of the supported platforms as follows:

1. Get a DXD Linux installer from the Support Portal (e.g., dirx91_9.7.382.1701.x86_lx-64.tar.gz).
2. Extract the installer to a temporary folder.
3. Open the **Dockerfile** in the root of the extracted installer.
4. Change the base image name in the first line according to the selected platform (e.g., from
   FROM opensuse/leap:15.6 AS base to
   FROM registry.suse.com/suse/sle15:15.6 AS base).
5. If the chosen platform uses a package manager other than zypper, update the **RUN zypper — non-interactive in gdb tar rsync** line accordingly.
6. Build the docker image using the modified **Dockerfile** according to the corresponding docker documentation: https://docs.docker.com/build. (e.g., docker image build -t dxd:9.7.138)

2.2. Content

The DirX Directory container image contains a full, standalone DirX Directory installation with all the server and client binaries. As in a standard Linux installation, the servers are automatically started by the **dirxdsas** process, which serves as the container entry point. The container image uses the UID 5000 and GUID 5000 to run the DirX Directory service.

2.3. Tagging

The DirX Directory container image is delivered with a single container image tag that contains the full version number; for example, **dxd:9.7.317**. By default, no other tag is provided. Additional tags can be created in the target system's image registry as necessary.

2.4. Initialization Scripts

By default, the DirX Directory container image does not contain an initialization script, but it does allow for the use of custom initialization scripts like other container images. These scripts can be used for database initialization or for executing any necessary initialization steps.

Any executable scripts mounted into the **/home/dirx/entrypoint-init.d/** folder are executed using the bash shell.The scripts are executed before the database is started, so any tools requiring exclusive access to the database (like **dbamboot** or **dirxload**) can be used, but scripts must not rely on any of the DirX Directory services to be running. For details on DirX Directory commands, see the chapter “DirX Commands” in the *DirX Directory Administration Reference*.

2.5. Environment Configuration

The safest and recommended way to set environment variables in a DirX Directory installation is to set them in the DIRX_INST_PATH**/conf/dirxenv.ini** configuration file. This is also the recommended way to set the environment for the DirX Directory container image. To work on the **dirxenv.ini** file, mount it to the **/home/dirx/conf/dirxenv.ini** path. For details on DirX Directory environment variables, see the chapter “Environment Variables” in the *DirX Directory Administration Reference*.

2.6. Networking

The DirX Directory container image uses the DirX Directory ports defined in the “IP Port Numbers” chapter of the *DirX Directory Administration Reference*. These ports can be exposed to the external network on demand.

2.7. Database Files

The DirX Directory container image only supports using a file-based DBAM database. Database files can be automatically created by implementing the logic in an initialization script, or they can be initialized by running a DirX Directory container image in which the database files can be created.

In addition to the database and translog files, the following files must also be persisted:

- **.DIRX_SyncFile** – this file is the synchronization interface between different DirX Directory tools and servers.
- **.DBAM_Profile** – this file stores the DBAM profiles used on the system.

As these files are read and modified by the servers and tools, they must be persisted using a volume with read and write permissions like the database files. By default, these files are stored in the **$DIRX_INST_PATH/server/conf** folder. However, in a containerized environment, configuration files and folders are usually mounted with read-only permissions. To solve this problem, you can use the **DIRX_SYNC_FILE_PATH** and **DIRX_DBAM_PROFILE_PATH** environment variables to configure the paths for these files so that they are stored in read-write locations, either with the DBAM database and translog files or on a separate read-write volume. See the environment configuration in the example DirX Directory Kubernetes project for an example.

3. Example Kubernetes Project

An example Kubernetes project that demonstrates the use and capabilities of the DirX Directory container image can be downloaded from the DirX support portal. The next sections describe the example project and how to use it.

3.1. Restrictions

This section describes restrictions that apply to the example DirX Directory Kubernetes project.

3.1.1. Distribution and Runtime

The example DirX Directory Kubernetes project is tested and supported only with the minikube Kubernetes distributions using the Docker Container runtime. Therefore, documentation is only provided for this scenario.

3.1.2. Artifacts

The example DirX Directory Kubernetes project is tested and supported only as is. Any Kubernetes configuration changes other than the procedures described in this document are outside the scope of DirX Directory support.

3.2. Prerequisites

Deploying and running the example DirX Directory Kubernetes project has the following prerequisites.

3.2.1. System Requirements

The host machine requires at least 8 GB RAM and 40 GB free disk space to deploy the DirX Directory Kubernetes example project.

3.2.2. Container Runtime

To run the example DirX Directory Kubernetes project, the Docker Container runtime must be installed and running on the host machine. Before any of the scenarios described below are executed, the following command must be able to run without errors on the host system:

docker container run --rm hello-world

Installation can be performed based on the distribution's official documentation or the Docker Engine documentation. See https://docs.docker.com for details.

3.2.3. Local Cluster

To run the example DirX Directory Kubernetes project, the host system must have a

configured minikube environment as described in the minikube installation instructions (see https://minikube.sigs.k8s.io/docs) and the minikube instance must be running. Use the following command to check it:

minikube status

The default-storageclass and storage-provisioner addons must be enabled. The status of the addons can be verified by executing the following command:

minikube addons list

3.3. Architecture

The example DirX Directory Kubernetes project runs the DirX Directory service in a stateful set Kubernetes resource connected with the necessary resources to provide network connection and persistence. The simplified architectural diagram of the example project is shown below.

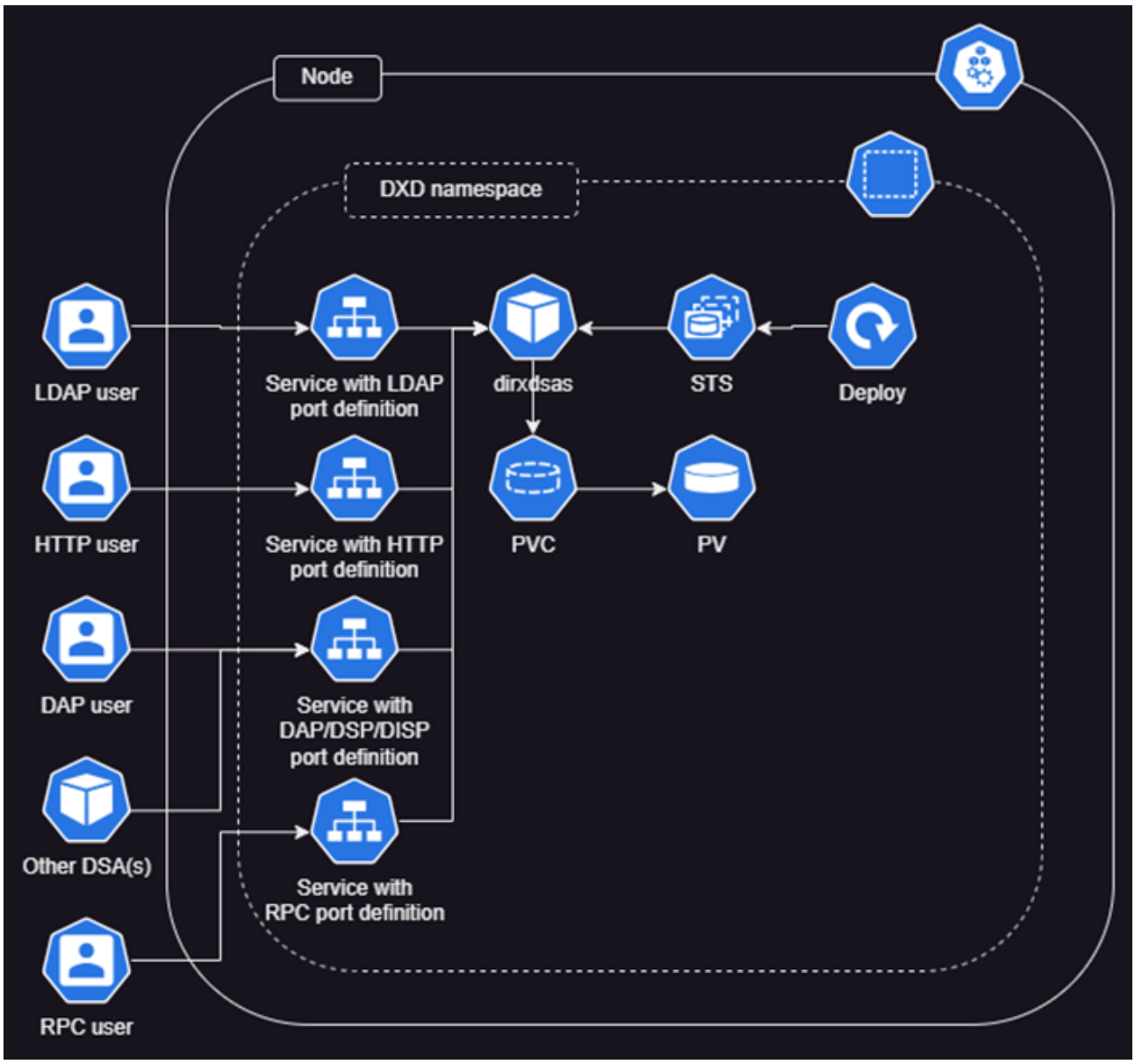

Figure 1. Example DirX Directory Kubernetes Project Architecture

3.4. Configuration

The root of the example DirX Directory Kubernetes project contains a set of folders that are similar to a normal DirX Directory installation: **client**, **conf**, **dsa**, **http**, **ldap**, **progsvr**, **tools** and several other configuration files.

The content of these folders corresponds to a normal DirX Directory installation and contains several types of configuration files. The next sections describe these files in more detail.

3.4.1. init_scripts_config_map.yaml

As in Kubernetes, the example DirX Directory Kubernetes project has a resource called an init container. The database initialization is implemented using this resource instead of the **entrypoint-init.d** scripts. The **init_scripts_config_map.yaml** configuration file contains the

initialization scripts to be executed before the DirX Directory container starts up. It handles initializing a database file and loading the **o=My-Company** example database. You can customize the **initialize_db.sh** script to set the DBAM profile to your requirements.

3.4.2. stateful_set.yaml

This configuration file contains the definition of the main stateful set resource. It defines the init container, the container, the container ports used and the mounted volumes. As the stateful set contains the definition of the container in use, the DirX Directory container image version to be used can be specified here. For this, the image property can be set in both the container and the init container definition.

3.4.3. service.yaml

The **service.yaml** configuration files define the service resources that are used to expose ports. There are multiple **service.yaml** files, one for each DirX Directory server. You can modify the exposed port numbers in these files.

3.4.4. *_pvc.yaml

These configuration files define the persistent volume claims that are used to persist the files used by the DirX Directory service. These files include the database files, log files, audit files, etc. You can adjust the size of the persistent volume claims by setting the storage parameter.

3.4.5. *config_map.yaml

Config maps are used to store the configuration files used by the DirX Directory service. There are several files with this naming structure, containing all configuration files used in a normal DirX Directory installation. You can modify these configuration files according to your requirements.

3.4.6. secret.yaml

Kubernetes secrets are used to store sensitive or binary data. In the example DirX Directory Kubernetes project, they are used to store, for example, client certificates, user certificates, the license files, etc. These secrets can be updated according to your requirements.

The **.pwd** files must be handled specially in **secret.yaml** configuration files. The DirX Directory services encrypt the **.pwd** files automatically. However, in Kubernetes, secrets are read-only resources and should not be modified once they are attached to a container. As a result, if the password file is set in the **secret.yaml** file without encryption, password file encryption will fail and the server will not be able to start. To solve this issue, the example DirX Directory Kubernetes project delivers an executable called **dirxencryptpwd** that can be used to encrypt the password. All **.pwd** files must contain the encrypted password.

3.5. Using the Example Kubernetes Project

This section describes example workflows you can use to experiment with the example

DirX Directory Kubernetes project.

3.5.1. Loading the Container Image

The first step of executing a containerized application is to get and load the container image. Download the DirX Directory container image from the DirX support portal and then load it to the local registry using the command:

```
minikube image load dxd-9.7.138.tar.gz
```

This command adds the **dxd:9.7.138** container image to the local registry inside the minikube environment. DirX Directory delivers the container image with only a single label containing the full version.

However, as the DirX Directory container image and the DirX Directory Kubernetes example project configuration files are delivered separately, the example project cannot refer to a specific DirX Directory version. So, in both the container and the init container, the image "dxd" is referred to with the default "latest" tag.

To use a specific image version, you can specify the tag in the image property of the containers and initContainers section of the stateful **set.yaml** configuration file. Alternatively, you can tag the image with the specific version tag to the default tag, as shown in the following command:

```
minikube image tag dxd:9.7.138 dxd
```

3.5.2. Starting the Example Project

The provided Kubernetes configuration files and the container image make it easy to run the DirX Directory service in the Kubernetes environment with the default My-Company configuration. To start the project:

- Get and load the DirX Directory container image as described in the section “Loading the Container Image”.
- Extract the Kubernetes configuration files and then modify the parameters defined in the configuration section to your requirements; for example, DirX Directory version, port numbers, storage space, and so on. When preparing the configuration files, it’s recommended to create a new namespace using the command:

  minikube kubectl — create ns *namespace*
- Apply the configuration files with the command:

  minikube kubectl — apply -n *namespace* **-Rf** *configuration_path*

  where *configuration_path* is the folder to which you have extracted the provided Kubernetes configuration yaml files.

- Check the status of the DirX Directory pod by running the command:

  minikube kubectl — get pods -n *namespace*

Here is an example startup command sequence for a DirX Directory service **dxd-standalone**:

```
minikube kubectl -- create ns dxd-standalone
minikube kubectl -- apply -n dxd-standalone -Rf kubernetes
minikube kubectl -- get pods -n dxd-standalone
```

Please note that starting up a pod takes time, so the last command may be repeated several times until the pod comes up, showing a Running status.

3.5.3. Tunneling the Ports

The example Kubernetes project uses LoadBalancer Kubernetes resources to expose its ports. By default, these ports are only available inside the minikube environment. To expose these ports to a specified bind address, use the command:

minikube tunnel --bind-address=*target_IP*

where *target_IP* is the IP address of the network interface to which the exposed ports should be bound. For example, to make the LDAP ports available outside minikube on the host's loopback interface, use the command:

```
minikube tunnel --bind-address=127.0.0.1
```

3.5.4. Customizing the Initialization

You can customize the example Kubernetes project by adjusting the delivered yaml configuration files and then applying the changes with the "apply" command described in the "Starting the Example Project" section.

Please note that the Kubernetes example project is tested and supported only as is. Kubernetes configuration changes other than the changes and procedures described in this document are outside the scope of DirX Directory support.

3.5.5. Executing Tools and Clients

The DirX Directory container image contains a full Linux installation. All tools and clients are included. You can use these tools by using kubectl's **exec** functionality. For example, you can open an interactive shell into the DirX Directory pod with the command:

minikube kubectl — exec -it -n *namespace* **dxd-0 — bash**

In the interactive shell, you can manage the DirX Directory service as a normal Linux

installation.

3.5.6. Deleting a Directory Instance

DirX Directory instances, if running in a separate namespace, can be deleted with the command:

minikube kubectl — delete ns *namespace*

Please note that this command will not clean up the persistent volumes. You should remove them manually.

3.5.7. Setting up Shadowing

This section describes how to implement a simple supplier-consumer scenario using the example Kubernetes project.

3.5.7.1. Create Master and Shadow Kubernetes Projects

As this shadowing scenario needs two DirX Directory instances with different configurations, the first step is to copy the example Kubernetes project files into two different folders called **dxd-master** and **dxd-shadow**. These folders will contain the necessary configuration for the supplier and the consumer respectively. To perform this task, run the following commands:

```
cp -R kubernetes dxd-master
cp -R kubernetes dxd-shadow
```

3.5.7.2. Set Master and Shadow DSA Identifiers

Next, set the **DIRX_HOST_NAME**, **DIRX_DSA_NAME** and **DIRX_OWN_PSAP** environment variables. Open the **dxd-master/conf/config_map.yaml** file and append the following rows to **dirxenv.ini**:

```
set DIRX_DSA_NAME=CN=DirX-k8s-master
set DIRX_HOST_NAME=dxd-service.dxd-master.svc.cluster.local
set
DIRX_OWN_PSAP=TS=DSA1,NA='TCP/IP_IDM!internet=1.2.3.4+port=1234',DNS=
'(HOST=dxd-service.dxd-master.svc.cluster.local,PLAINPORT=21200)'"
```

Now open the **dxd-shadow/conf/config_map.yaml** file and append the following rows to **dirxenv.ini**:

```
set DIRX_DSA_NAME=CN=DirX-k8s-shadow
set DIRX_HOST_NAME=dxd-service.dxd-shadow.svc.cluster.local
```

```
set
DIRX_OWN_PSAP=TS=DSA2,NA='TCP/IP_IDM!internet=1.2.3.4+port=1234',DNS=
'(HOST=dxd-service.dxd-shadow.svc.cluster.local,PLAINPORT=21200)'"
```

3.5.7.3. Adjust Duplicate LoadBalancer Resources

As both **dxd-master** and **dxd-shadow** are copied from the same source, they contain the same definitions for the **LoadBalancer** service resources. If multiple **LoadBalancer** resources are started with the same port, it will result in port collisions when the traffic is tunneled to an IP address. So as the next step, you should delete or modify these resources. If you would like to access only one of the nodes from an external IP, you can delete the **dsa/service.yaml**, **ldap/service.yaml**, **progsvr/service.yaml** and **http/service.yaml** files from one of the copied folders. If you would like to access both, you can modify the port numbers in these files to avoid port collision.

3.5.7.4. Start Master and Shadow Instances

When the configuration is finished, start the **dxd-master** instance with the following commands:

```
minikube kubectl -- create ns dxd-master
minikube kubectl -- apply -n dxd-master -Rf dxd-master/
minikube kubectl -- get pods -n dxd-master
```

Then start the dxd-shadow instance:

```
minikube kubectl -- create ns dxd-shadow
minikube kubectl -- apply -n dxd-shadow -Rf dxd-shadow/
minikube kubectl -- get pods -n dxd-shadow
```

3.5.7.5. Create the Shadowing Agreement

After these commands are executed, there are two separate standalone DirX Directory services running. Both are loaded with the **o=My-Company** example database. The next step is to create the shadowing agreement between the two standalone DSAs using the following commands:

```
minikube kubectl -- exec -it -n dxd-master dxd-0 bash
dirxadm -c "defbind; sob create -consumer {/CN=DirX-k8s-shadow} \
    -agreementid 15 \
    -consumerpsap
{TS=DSA2,NA='TCP/IP_IDM!internet=1.2.3.4+port=1234',DNS='(HOST=dxd-
service.dxd-shadow.svc.cluster.local,PLAINPORT=21200)'} \
```

```
    -supplier {/CN=DirX-k8s-master} \
    -supplierpsap
{TS=DSA1,NA='TCP/IP_IDM!internet=1.2.3.4+port=1234',DNS='(HOST=dxd-
service.dxd-master.svc.cluster.local,PLAINPORT=21200)'} \
    -consumerkind CENTRALADMIN  \
    -status cooperative \
    -agreement {SS={AREA={CP={/o=My-Company},\
        RA={DEF=TRUE}},\
        ATT={DEF=TRUE}},\
        UM={SI={OC=TRUE}},CHANGEO=FALSE} \
    -pol {CONS={REPLS=TRUE}}"
```

3.5.8. Setting up Crash Handling

By default, the DirX Directory watchdog (**dirxdsas**) handles all server crashes. However, the minikube environment does not allow **ptrace** calls by default, so the watchdog procedure is not allowed to generate core dumps.

To activate automatic core dump collection, install the **systemd-coredump** package and then disable the watchdog's core dump handling by setting **DIRX_WDOG_CRASH_HANDLER=0** in **dirxenv.ini**. These settings enable **systemd** to collect the core dumps.

3.6. Troubleshooting

All files (including log, audit, ldif, and others) in the DirX Directory container image are written as they are in a normal Linux installation, so files required for troubleshooting are written as files. In the example Kubernetes project, all paths used to store these files are mapped to a persistent volume claim provided by minikube's hostpath-provisioner so that files needed for troubleshooting are persisted. The provisioner creates the persistent volumes in the **/var/lib/docker/volumes/minikube/_data/hostpath-provisioner/***namespace* folder. The log files of the currently running instance can be found in the separate folders. However, the logs of previous instances are copied to a special log folder called **dxd-log-persistence-pvc**. The log persistence folder is automatically cleaned up by deleting the log of all pods written more than 30 days ago.

DirX Product Suite

The DirX product suite provides the basis for fully integrated identity and access management; it includes the following products, which can be ordered separately.

DirX Identity

DirX Identity provides a comprehensive, process-driven, customizable, cloud-enabled, scalable, and highly available identity management solution for businesses and organizations. It provides overarching, risk-based identity and access governance functionality seamlessly integrated with automated provisioning. Functionality includes lifecycle management for users and roles, cross-platform and rule-based real-time provisioning, web-based self-service functions for users, delegated administration, request workflows, access certification, password management, metadirectory as well as auditing and reporting functionality.

DirX Directory

DirX Directory provides a standards-compliant, high-performance, highly available, highly reliable, highly scalable, and secure LDAP and X.500 Directory Server and LDAP Proxy with very high linear scalability. DirX Directory can serve as an identity store for employees, customers, partners, subscribers, and other IoT entities. It can also serve as a provisioning, access management and metadirectory repository, to provide a single point of access to the information within disparate and heterogeneous directories available in an enterprise network or cloud environment for user management and provisioning.

DirX Access

DirX Access is a comprehensive, cloud-ready, scalable, and highly available access management solution providing policy- and risk-based authentication, authorization based on XACML and federation for Web applications and services. DirX Access delivers single sign-on, versatile authentication including FIDO, identity federation based on SAML, OAuth and OpenID Connect, just-in-time provisioning, entitlement management and policy enforcement for applications and services in the cloud or on-premises.

DirX Audit

DirX Audit provides auditors, security compliance officers and audit administrators with analytical insight and transparency for identity and access. Based on historical identity data and recorded events from the identity and access management processes, DirX Audit allows answering the “what, when, where, who and why” questions of user access and entitlements. DirX Audit features historical views and reports on identity data, a graphical dashboard with drill-down into individual events, an analysis view for filtering, evaluating, correlating, and reviewing of identity-related events and job management for report generation.

For more information: support.dirx.solutions/about

Eviden is a registered trademark © Copyright 2025, Eviden SAS – All rights reserved.

Legal remarks

On the account of certain regional limitations of sales rights and service availability, we cannot guarantee that all products included in this document are available through the Eviden sales organization worldwide. Availability and packaging may vary by country and is subject to change without prior notice. Some/All of the features and products described herein may not be available locally. The information in this document contains general technical descriptions of specifications and options as well as standard and optional features which do not always have to be present in individual cases. Eviden reserves the right to modify the design, packaging, specifications and options described herein without prior notice. Please contact your local Eviden sales representative for the most current information. Note: Any technical data contained in this document may vary within defined tolerances. Original images always lose a certain amount of detail when reproduced.